Chapter 2
Low-Power Circuits: A System-Level Perspective

Youngsoo Shin

Abstract Popular circuit techniques for reducing dynamic and static power consumption are reviewed. The emphasis is on the implication when they are applied, e.g., area increase, because this may serve as important information during system-level design. The estimation of power and temperature is also reviewed.

2.1 Introduction

During the architectural design (or system-level design, broadly speaking), a lot of what-if questions are likely to be raised and answered. For example, in a network processor, the designers may consider employing two Ethernet controllers, instead of a single one, to improve throughput, but may also want to validate the choice in terms of chip area [1].

Due to the growing importance of power consumption, it is now tempting to assess the design choice in terms of power: what happens if clock gating is applied to block A, which contains many synchronous memory elements; what happens if body biasing is used in block B, which stays in standby mode for most of its operation time? These questions should be answered after the implication of applying each circuit technique is precisely understood from a system-level perspective; e.g., how much does the circuit area increase when clock gating is applied to A, and what is the latency to put B in standby mode and bring it back to active mode?

This chapter is organized to review various low-power circuit techniques from a system-level perspective. A technique to estimate power consumption is discussed in Sect. 2.3; thermal analysis, which has become very important, is also addressed. Power consumption can be categorized into dynamic power during operational time and static power during standby periods. Representative circuit techniques to reduce the dynamic component, i.e., clock gating and dual-V_{dd}, as well as other techniques are reviewed in Sect. 2.4. Techniques to reduce the static component, such as power gating and body biasing, are presented in Sect. 2.5.

Y. Shin (✉)
KAIST, Daejeon, Republic of Korea
e-mail: youngsoo@ee.kaist.ac.kr

C.-M. Kyung, S. Yoo (eds.), *Energy-Aware System Design*,
DOI 10.1007/978-94-007-1679-7_2,

2.2 CMOS Power Consumption

To understand the nature of power consumption of CMOS circuits, consider the chip floorplan illustrated in Fig. 2.1. The overall operation of a floorplan block can be classified as being in active or in standby mode. Active mode refers to the period of time when the block is actively computing to produce valuable output; the remaining period is called standby mode. In active mode, there are two components of power consumption: dynamic and static power. *Dynamic power* is consumed while a transistor is switching. The length of time that it switches is usually a small proportion of a clock cycle; for the remaining time, the transistor consumes *static power*. Standby mode, which does not involve any transistor switching, consists of static power alone (assuming that there is also no switching activity in a clock). It is important to understand that the static power in active mode is a transient one, while that in standby mode is a static one; therefore, their amounts are very different, as we address later in this section.

2.2.1 Dynamic Power

While the output of the CMOS inverter shown in Fig. 2.1 makes a pair of rising and falling transitions, the amount $C_{\mathrm{L}} V_{\mathrm{dd}}^2$ of energy is dissipated, half of it by the pMOS transistor and the other half by the nMOS transistor. C_{L} is the load capacitance, which models the gate capacitance of fanout gates, the wire capacitance, and the intrinsic capacitance of the inverter itself.

The average power consumption due to the switching, which is the total energy dissipation during a particular period of time divided by the length of that period, is given by the well-known expression

$$P_{\mathrm{sw}} = \alpha C_{\mathrm{L}} V_{\mathrm{dd}}^2 f, \tag{2.1}$$

where f is the clock frequency and α[1] is the probability of the output making a pair of rising and falling transitions in a single clock cycle. Note that $\alpha \leq 0.5$ for any combinational gate unless there is a glitch; in practical circuits, α turns out to be very low, typically less than 0.05. Gates that are driven by a clock, for example those in clock buffers, have $\alpha = 1.0$.

Another component of dynamic power consumption, denoted by P_{sc}, is caused by *short-circuit current*. This is the current that flows while both the nMOS and pMOS transistors are turned on for a short period of time when the input signal makes a transition (from 0 to 1 or 1 to 0). Interestingly, P_{sc} decreases with increasing C_{L} [2], because the output, which changes its value more slowly when heavily loaded, keeps the short-circuit current from increasing. C_{L}, however, cannot be arbitrarily increased due to increased circuit delay. P_{sc} is usually pre-characterized

[1] Some people use α as a probability that the output makes a transition (either rising or falling) rather than a pair of transitions. With this definition, $1/2\alpha$ would be used instead of α in (2.1).

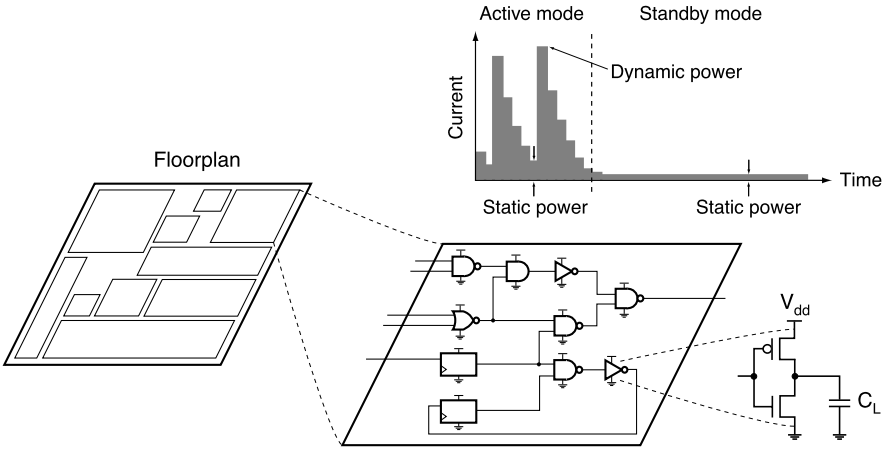


Fig. 2.1 Power consumption of an architectural block

when each gate is designed and is available during power estimation. In practical circuits, P_{sc} is a small proportion of the total dynamic power consumption P_{dyn}; e.g., P_{sc}/P_{dyn} is estimated to be about 10% [3].

2.2.2 Static Power

The static power consumption is a result of the device leakage current, which originates from various physical phenomena [4]. Three components of leakage (subthreshold, gate tunneling, and junction leakage) get more attention than the other ones due to their large proportion in the total static power. The relative importance of these components differs with the technology, the temperature, the style of the circuit, and so on. For instance, gate leakage is important in static random access memory (SRAM) circuits since they typically rely on devices of larger gate length to reduce random dopant variations, while subthreshold leakage is dominant in logic circuits [5].

The subthreshold leakage occurs when the gate-to-source voltage of a transistor is below its threshold voltage (V_{th}), i.e., when a device is presumed to be turned off. It is well known that this leakage component increases exponentially with decreasing V_{th}, increasing temperature, and increasing gate-to-source voltage. This implies the growing importance of subthreshold leakage as CMOS technology scales down, since V_{th} tends to decrease to maintain circuit speed. It also implies that any quantitative result on static power should be carefully understood; e.g., the value may be very different for different temperatures.

The standby leakage (leakage in standby mode) of the 2-input NAND gate shown in Fig. 2.2(a) for the different inputs is given in the second column of Table 2.1. It is well known that this leakage is lowest when the input is 00, as Table 2.1 confirms.

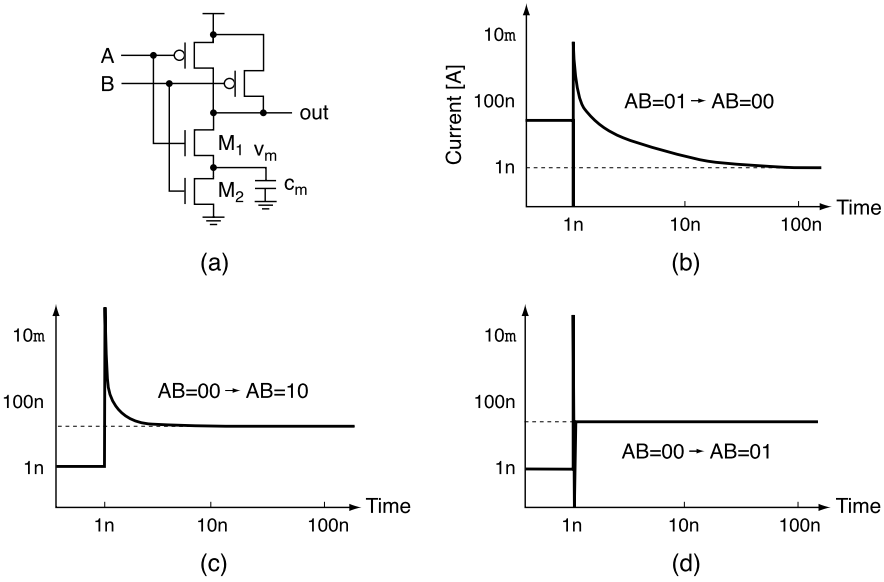


Fig. 2.2 (**a**) A 2-input NAND gate: active leakage for input transitions (**b**) from 01 to 00, (**c**) from 00 to 10, and (**d**) from 00 to 01

Table 2.1 Standby and active leakage of a 2-input NAND gate in 45-nm technology

Input (AB)	Standby leakage (nA)	Active leakage (nA)		
		1 ns	5 ns	10 ns
00	1.0	11.2	3.0	1.9
01	16.6	16.2	16.6	16.6
10	10.3	17.7	10.4	10.3
11	26.4	26.4	26.4	26.4

The reason is that there is a positive voltage v_m which builds up between M_1 and M_2 and turns M_1 off strongly, due to a negative gate-to-source voltage; this voltage also raises the effective threshold voltage of M_1. The whole phenomenon is called the stacking effect [6], because the leakage shrinks as stacked MOS transistors are turned off.

We now turn our attention to active leakage (leakage in active mode). When the input is maintained at 01, the internal node capacitance c_m is fully discharged. If the input is changed to 00 after 1 ns, as depicted in Fig. 2.2(b), the small leakage current through M_1 starts to charge c_m. As v_m rises, the leakage through M_1 falls further due to the stacking effect. But this transition takes a long time, as shown in Fig. 2.2(b). The effect on leakage of a change of input from 00 to 10 is shown in Fig. 2.2(c). The large turn-on current through M_1 initially charges c_m; however, as v_m rises, M_1 turns off, but then its leakage current takes over and continues to charge c_m, even though the leakage is gradually falling. If M_2 is turned on, for instance by

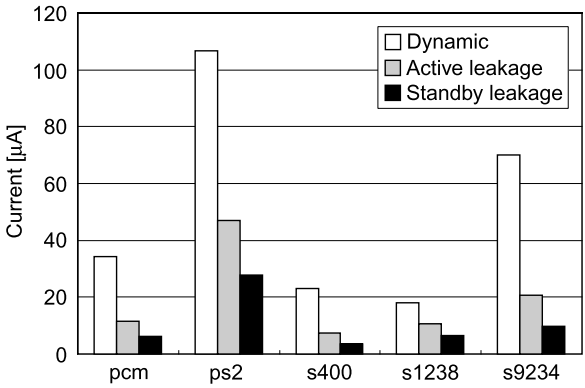


Fig. 2.3 Comparison of three components of power consumption in 45-nm technology; leakage is measured assuming 125°C

the change of input from 00 to 01 shown in Fig. 2.2(d), the corresponding leakage transition is virtually spontaneous since c_m is quickly discharged.

The average active leakage over different periods after the change of input value is given in the last three columns of Table 2.1. Each value is also averaged over all the transitions that lead to the inputs shown in the first column: thus the first row covers transitions from 01 to 00, from 10 to 00, and from 11 to 00.

The standby and the active leakage are about the same when a 1 is applied to input B (01 and 11 of Table 2.1), which turns on M_2. The leakages for 10 and, especially, 00, are significantly different, particularly for the period immediately after the transition, implying a higher operating frequency.

2.2.3 Analysis

There are now three components of power consumption: dynamic, active leakage, and standby leakage. The first two components are sources of active-mode power consumption; the last defines standby-mode power consumption. Experiments were performed in 45-nm technology to understand the relative measure of the components; the results are shown in Fig. 2.3. Example circuits were taken from International Symposium on Circuits and Systems (ISCAS) benchmarks as well as from OpenCores [7]. The current was obtained by applying 100 random vectors; the clock period was arbitrarily assumed at 5 ns.

Active leakage represents, on average, 28% of the total active-mode power consumption; it is as high as 37% in s1238 and as low as 23% in s9234. Note that the leakage was measured in conditions where it becomes as large as possible, i.e., a fast process corner in which V_{th} is smallest and at the highest operating temperature. Since dynamic power is scarcely affected by these parameters, the proportion of active leakage will become smaller in different conditions. For example, its proportion decreases to 14% in a nominal process corner with the same temperature.

The average standby leakage is 54% of the average active leakage. The variation in the leakage ratio between circuits can be explained by the extent of the stacking

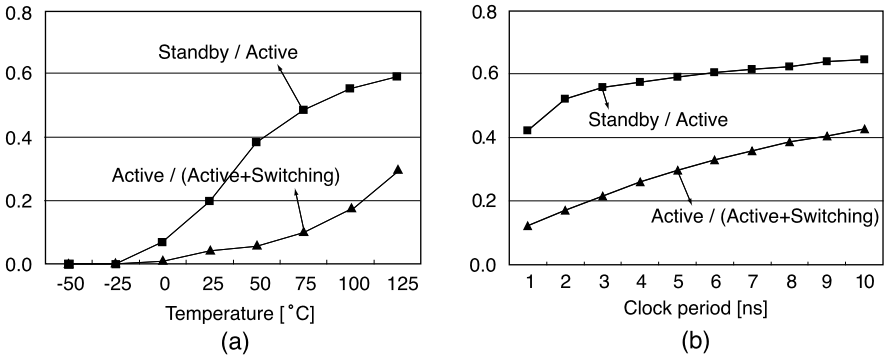


Fig. 2.4 Ratio of standby to active leakage, and the proportion of active leakage in circuit ps2: with (**a**) varying temperature and (**b**) varying clock period

effect in each circuit. When there are more gates that exhibit the stacking effect in standby mode, we expect the difference between active and standby leakage to increase. This can be confirmed by counting the number of inverters and flip-flops, which are representative of the gates without the stacking effect.

The proportion of active leakage decreases with temperature, as shown in Fig. 2.4(a). The ratio of standby to active leakage also declines, as Fig. 2.4(a) shows, suggesting that the importance of active leakage grows as the temperature drops. When this happens, the transient change in active leakage due to a transition (see Fig. 2.2) takes longer because of its reduced magnitude, which means that c_{m} is charged more slowly: this increases the difference between active and standby leakage.

As the clock frequency increases and the clock period decreases, the magnitude of the active leakage will increase while the standby leakage remains the same. This is evident from the decreasing ratio between the standby and active leakage shown in Fig. 2.4(b). The total switching current is independent of the clock period, as long as that period is sufficient to accommodate all the switching required. While the average switching current and the active leakage both increase as the clock period decreases, the average switching current increases more rapidly. Thus, the active leakage comes to represent a lower proportion of the total active-mode current, as we see in Fig. 2.4(b).

The contribution of the three components in energy dissipation is determined by the amount of time for which each component is responsible. This in turn is dependent on the fraction of time a circuit stays in active mode, i.e., the duty cycle D. Let dynamic power and active leakage be 72% and 28% of the active-mode power consumption, respectively, and active leakage be 1.87 times the standby leakage. Figure 2.5 illustrates the contribution of the three components with different values of D, e.g., $4.80D/(5.67D+1)$ for dynamic power. When $D = 0.1$ such as in a cell phone, 58% of the energy is due to standby leakage, while 31% and 11% are due to dynamic and active leakage. It is apparent that most of the energy is dissipated by dynamic power as D increases, which arises in stationary devices such as servers.

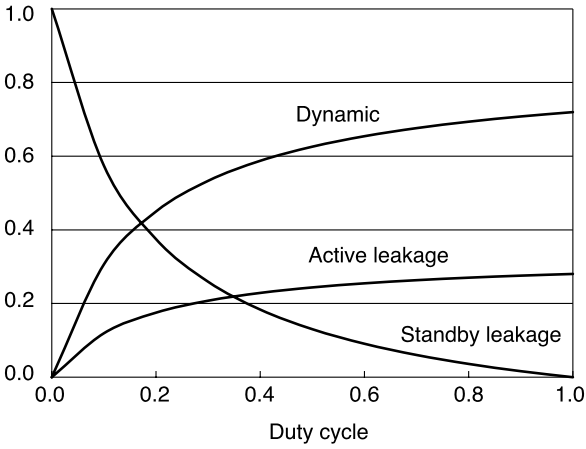


Fig. 2.5 Contribution of three components in energy dissipation with varying duty cycle D

2.3 Estimation of Power Consumption

The biggest part of answering what-if questions during architectural design is the ability to estimate power consumption, before and after a particular circuit technique is applied; this is a subject of this section. We also address temperature estimation because the main quantity that determines temperature is power consumption and because temperature has become a roadblock in technology scaling.

2.3.1 Dynamic Power

Expression (2.1) suggests that the estimation of P_{sw} comes down to estimating α of each node, once C_L is extracted. This is done either by simulation or by probabilistic analysis.

Different gate delay models can be used in a simulation approach. The simplest model assumes zero gate delay for the sake of simulation time. Each gate can have at most one transition per input vector, since all transitions occur at the same time. If real delay is used, each gate may have different delay resulting in different arrival times at the gate inputs, which causes more than one transition per input vector. But, this takes more time than simulation under zero delay. Gate-level simulation is reported to yield an error of $\pm 15\%$ compared to circuit-level simulation, which exhibits $\pm 5\%$ error [8]. Another issue is the preparation of input vectors. This is either done by designer-specified use scenarios, or is based on generating a sequence of random vectors. The interesting question here is the number of vectors that should be provided for reasonable accuracy. Experimental study [8] states that using any 100 or 10 consecutive vectors guarantees an error within $\pm 5\%$ or $\pm 15\%$ (compared to using the whole sequence of vectors from use scenarios), which implies that 10 should be enough for the accuracy of gate-level simulation.

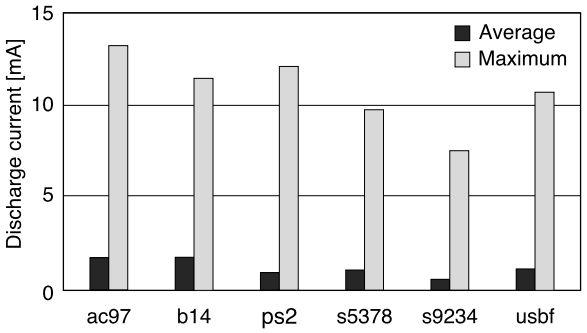


Fig. 2.6 Comparison of average and maximum discharge current

Probabilistic analysis is more convenient from a designer's perspective. One may simply assume a signal probability (probability of signal being at 1) at each circuit input, quite often 0.5. The probabilities are then propagated toward circuit outputs. The propagation of independent signals is straightforward; e.g., the signal probability at the output of an AND gate is 0.25 when the signal probability of both inputs is 0.5. However, in general circuits, many signals are not independent due to reconvergent fanout; i.e., the same fanout converges at the same gate after going through different paths to the gates. The propagation in this case becomes more difficult, although several methods have been proposed [9].

Note that these power estimation methods target average power consumption. The maximum power consumption, which is necessary for designing a power distribution network, is significantly larger than the average one. This is quantitatively shown for several circuits in Fig. 2.6, in which the difference ranges from 6 to 7 times.

Accuracy of Estimation The important issue in power estimation is its accuracy. This is affected by several factors such as delay model, wire model, and test vectors, but, more importantly, by the design stage in which power estimation is performed. During system-level design, many blocks are in a register transfer level (RTL) description. The description then goes through logic synthesis, in particular technology mapping, to obtain a technology-mapped netlist; some optimizations are then performed, and the layout is finally obtained. Before layout design, the inaccuracy of power estimation ranges $\pm 15\%$; a similar inaccuracy is observed in power estimation before optimization. However, the error of power estimation before technology mapping (an estimation without actual netlist) can reach a factor of 4 or 10, which invalidates any estimation effort at that early stage [8].

2.3.2 Static Power

For a given gate-level netlist, estimating leakage power is generally more difficult than estimating switching power. Switching power is weakly dependent on device

parameters and operating environments. However, leakage power is strongly affected by the variations of process parameters (e.g., gate length, oxide thickness, and channel dose), variations of operating environment (temperature and V_{dd}), and different input patterns.

The dependency of leakage current on process variations is the strongest; e.g., for 3σ die-to-die V_{th} variation of 30 mV in 180-nm CMOS technology, the leakage current can vary by a factor of 20, while the frequency varies only by 20% [10]. Die-to-die variations are typically taken into account by using process corners; i.e., we can estimate leakage current by assuming one particular set of deterministic device parameters. However, within-die variations, which are occupying an increasing proportion of total process variations with technology scaling, can only be captured by statistical estimation. The dependency of leakage on operating environments is also strong, although less strong than for process variations in practice. Leakage has a superlinear dependency on temperature, e.g., a 30°C change of temperature causes leakage to increase by 30%, and its dependency on supply voltage is exponential, e.g., a 20% fluctuation of V_{dd} causes leakage to change by a factor of 2 or more [11]. Therefore, for accuracy, leakage estimation should be coupled with an analysis of temperature and V_{dd} distribution. The dependency of leakage on input vectors is strong in individual gates, but becomes very weak in whole circuits, especially as circuits have more levels due to lack of controllability.

Static Estimation For leakage analysis or simulation, each gate in the library must be characterized in its leakage. For example, for a 2-input NAND gate, the leakage for each input combination can be characterized: L_{00}, L_{01}, L_{10}, L_{11}, where L_{ij} indicates leakage when the inputs take i and j. Alternatively, for simplicity, its leakage could be characterized by the average value.

If the leakage of all the gates is characterized, the leakage of an individual gate can be obtained if we know the signal probability of each input. For example, the leakage of the 2-input NAND gate is given by $(1-p_1)(1-p_2)L_{00} + (1-p_1)p_2L_{01} + p_1(1-p_2)L_{10} + p_1p_2L_{11}$, where p_1 and p_2 are the signal probabilities of two inputs. The leakage of the whole circuit can then be obtained by summing all the leakages. Thus, the key step is to derive the signal probability of all internal nodes given the signal probability of the primary input, which is the same process as in dynamic power estimation.

Statistical Estimation There are two methods to incorporate within-die process variation in leakage analysis: Monte Carlo simulation (simulation with repeated random sampling of variation source) or statistical estimation. Figure 2.7 illustrates typical leakage histograms after Monte Carlo simulation with 45-nm technology [12], in which σ of V_{th} is assumed to be 10% of its normal value. The histogram roughly follows a lognormal distribution.

In statistical estimation, the leakage of each gate is modeled as a lognormal, i.e., αe^{Y_i} [13], where Y_i is a function of process parameters such as gate length and gate oxide thickness, and approximated as a normal distribution. It is shown that both subthreshold and gate tunneling leakage follow this model. The full-chip leakage is then a sum of lognormals, which can be approximated as another lognormal

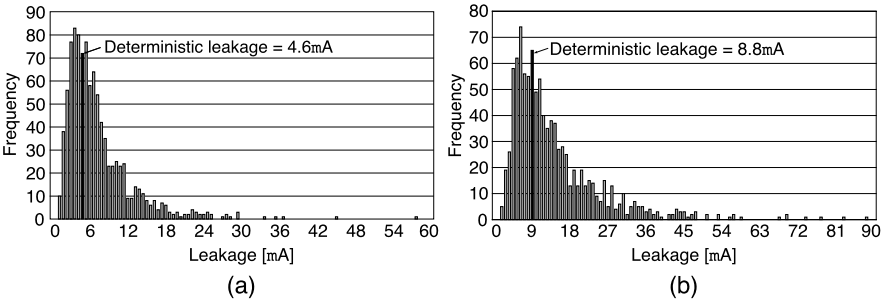


Fig. 2.7 Monte Carlo simulation of leakage: (**a**) c432 and (**b**) c1350

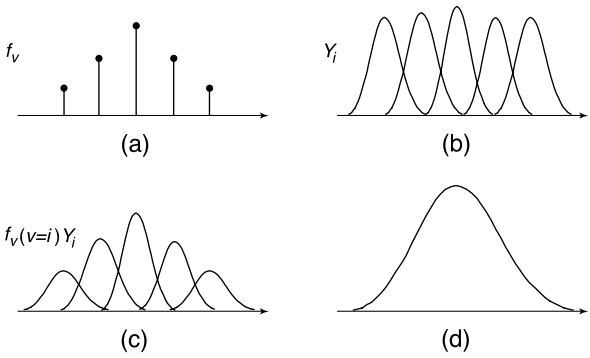


Fig. 2.8 Statistical leakage estimation considering both D2D and WID variations: (**a**) discrete sample of D2D variation, (**b**) Y_i at different instances of D2D variation, (**c**) Y_i scaled by the probability of D2D variation, and (**d**) the aggregate leakage

or, more accurately, as an inverse-gamma distribution [14]. If leakage is estimated from a layout, a spatial correlation of device parameters must be taken into account. In other words, Y_i and Y_j are highly correlated if gates i and j are closely located. A chip is divided into an imaginary grid, and a correlation coefficient is defined between a pair of grids, which is then incorporated into the leakage estimation [13].

Statistical leakage estimation considering both die-to-die (D2D) and within-die (WID) variations can be done, as illustrated in Fig. 2.8 [15]. D2D variation is sampled at discrete points (a). Each sampled value becomes a mean of a corresponding normal distribution of Y_i (b). Each Y_i is scaled by the corresponding probability of the sample from D2D space (c). Statistical leakage estimation is done for each Y_i and aggregate leakage is obtained (d).

2.3.3 Temperature Estimation

Temperature changes because of the convection of heat. Therefore, it is reasonable to expect to produce temperature change by adjusting the location of hotter and colder blocks, i.e., by trying different floorplans. It is reported that different floor-

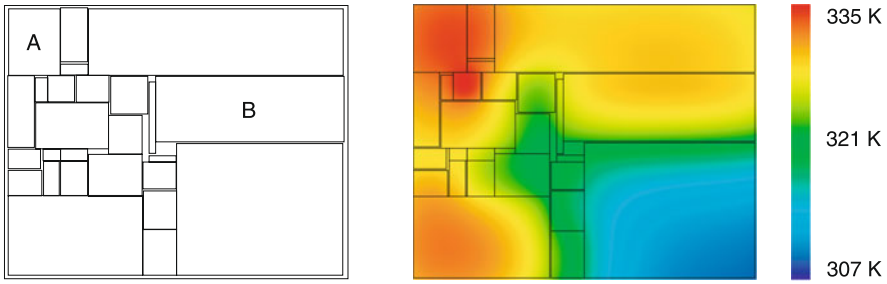


Fig. 2.9 (**a**) Floorplan of an example chip and (**b**) thermal map

plans of microprocessors can yield a difference of maximum temperature of as high as 37°C [16].

For a given floorplan, as shown in Fig. 2.9(a), the following heat conduction equation is solved to generate a thermal map, shown in Fig. 2.9(b):

$$\rho C_{\mathrm{p}} \frac{\partial T(x,y,z,t)}{\partial t} = \nabla\big[\kappa(x,y,z,t)\nabla T(x,y,z,t)\big] + g(x,y,z,t), \tag{2.2}$$

where T is the temperature which we try to obtain, g is the power density of a heat source, and κ denotes thermal conductivity; ρ and C_{p} are material-dependent parameters. Physically, (2.2) implies that the energy stored in a volume V (left-hand side) is equal to the sum of the heat entering V through its boundary surface and the heat generated by itself (right-hand side).

In general, steady-state temperature is of importance because, once a chip reaches that state, the temperature does not respond to an instantaneous change of power consumption. This is due to the relatively large time constant of heat conduction (a few milliseconds) compared to that of a clock cycle (some picoseconds). In a steady state, in which there is no change of temperature over time, the following equation can be solved:

$$\nabla^2 T(x,y,z) = -g(x,y,z)/\kappa, \tag{2.3}$$

where κ is approximated to be constant. Note that g is typically given for each block, say A and B of Fig. 2.9(a); in other words, we approximate the power density of A to be homogeneous—this can be a source of error when the block is very big. Average power consumption (over some period of time) is used for g of (2.3), which can be another source of error, particularly when we try to obtain the maximum temperature. These limitations should be kept in mind when temperature is referred to after estimation.

There are several methods to solve (2.2) or (2.3). Numerical methods include the finite difference method (FDM) or finite element method (FEM), both of which discretize the continuous space domain into a finite number of grid points. But these methods are very slow, usually taking tens or hundreds of minutes; thus, it is not practically possible to use them in any optimization loop.

Fast estimation methods do exist. A notable one is to use a *thermal RC circuit* [17]. This is a circuit built based on the analogy between heat transfer and electrical current: heat flow can be described as a current flowing through a thermal resistance, thus yielding a temperature difference analogous to voltage. Thermal resistance and capacitance are modeled on a per-block basis or, more accurately, on a per-grid basis, in which a chip is divided into a number of imaginary grids. Another fast method to solve (2.3) is to use a Green's function. It can be readily shown that (2.3) is equivalent to

$$T(\mathbf{r}) = \int_{-\infty}^{\infty} G(\mathbf{r}, \mathbf{r}_0) \left(-\frac{g(\mathbf{r}_0)}{\kappa} \right) d\mathbf{r}_0, \tag{2.4}$$

where $\mathbf{r}$ is (x, y, z) and $\mathbf{r}_0$ is a particular value of $\mathbf{r}$. G satisfies $\nabla^2 G(\mathbf{r}, \mathbf{r}_0) = \delta(\mathbf{r} - \mathbf{r}_0)$ and is called a Green's function; i.e., G is a Green's function if its Laplacian is a delta function. Instead of solving partial differential equation (2.3), we can use (2.4) to directly give T once G is known. The product of cosine functions [18] and the division of hyperbolic functions have been used for G.

2.4 Circuits to Reduce Dynamic Power

Many circuit techniques have been proposed to reduce dynamic power consumption. Two of them, namely clock gating and dual-V_{dd}, deserve attention because of their popularity and effectiveness, and are reviewed in this section in detail. Other techniques are summarized in Sect. 2.4.3.

2.4.1 Clock Gating

It is well known that a clock distribution network takes a large portion of total power consumption, e.g., 18% to 36% for processors and 40% for ASICs [19]. This is because the elements of the network including flip-flops (or latches) and clock buffers, as shown in Fig. 2.10, are always triggered. A simple way to reduce this consumption is to gate the clock to a flip-flop, say A, when its input and output are the same. If a clock to A and B can be gated at the same time, we may try to gate the buffer C instead, or higher stage buffers if more flip-flops can be gated together.

Conceptually, clock gating can be implemented as shown in Fig. 2.11(a). The block called clock gating logic determines when the combinational logic does not perform its computation (EN = 0) and when it does (EN = 1). Two things should be noted in regard to clock gating logic. It is an extra logic, which causes an increase of circuit area and power consumption; it therefore should be kept small as much as possible. Clock gating logic itself is a combinational logic, and it thus may generate a hazard; in particular, a static 1-hazard (a change of logic value from

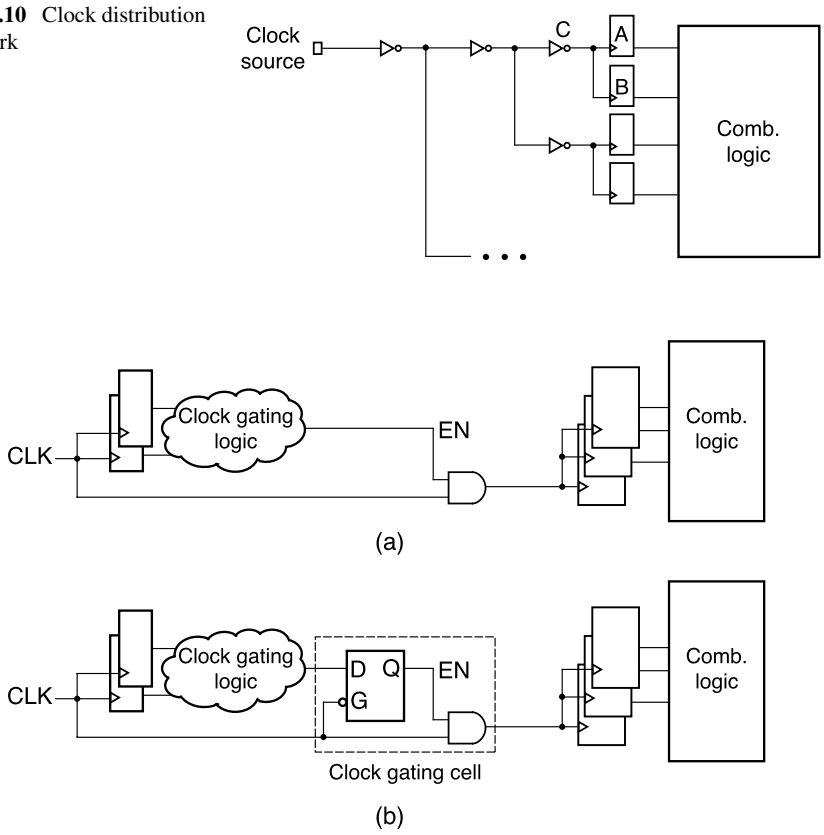


Fig. 2.10 Clock distribution network

Fig. 2.11 Clock gating: (**a**) concept and (**b**) implementation

1 to 0 and back to 1, for a short period of time) while CLK = 1 makes the flip-flops capture their inputs when they are not supposed to. This is resolved by using a negative sensitive latch, as shown in Fig. 2.11(b). When CLK = 1, the latch is opaque and thus blocks any hazard from clock gating logic. The latch together with an AND gate are typically called a clock gating cell. Note that a positive sensitive latch and an OR gate are used if the flip-flops are falling edge triggered ones.

From the designer's perspective, the challenge is to design the clock gating logic such that flip-flops are gated as often as possible while the gating logic is kept small. This is done either manually by human designers or automatically by CAD tools. A generic form of digital circuit consists of a data path and controller, as illustrated in Fig. 2.12. Designers should know when each functional unit is idle from a scheduled data flow description, which could guide them to design clock gating logic. The controller is typically modeled as a finite state machine (FSM) such as the one shown in Fig. 2.12; self-loops associated with states A and B correspond to the mo-

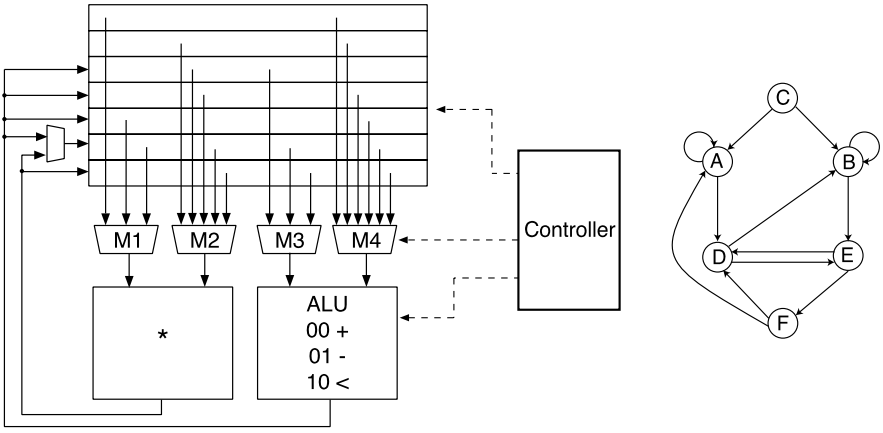


Fig. 2.12 Digital circuit consists of data path and controller

ment when the clock can be gated; e.g., if state A is assigned binary representation 000, the register content does not change before and after the self-loop.

Automatic synthesis of clock gating logic is a bottom-up approach. Let $\delta_i(\cdot)$ be the Boolean expression for the input of the ith flip-flop and let S_i be its output. The clock can be gated when these two take the same value (either 1 or 0); the clock gating logic is thus given by

$$g_i = \overline{\delta_i(\cdot) \oplus S_i}. \tag{2.5}$$

Implementing each g_i as a separate logic incurs too much overhead. Some g_i's thus should be merged as much as possible. Note that if g_1 and g_2 are merged, flip-flops 1 and 2 are gated only when both can be gated, thereby reducing the gating probability. Therefore, merging g_i's is a trade-off between clock gating logic and gating probability. There are other techniques to reduce the complexity of clock gating logic, e.g., using don't cares, logical approximation, and so on [20].

Figure 2.13 shows the result of clock gating synthesis using a commercial CAD tool [21]. Interestingly, the area starts to decrease as clock gating is applied to some flip-flops, less than 400 flip-flops in Fig. 2.13. This happens because a feedback multiplexer, which is attached to a flip-flop to retain its current value when it needs to, can be removed when clock gating is applied; i.e., if the clock is gated, the value of the flip-flop is retained anyway. This benefit is outweighed by an increasing amount of clock gating logic as more flip-flops are gated. The power consumption monotonically decreases, indicating that extra power consumption from clock gating logic is not large enough (due to the low switching activity of combinational logic) to mask the reduced power consumption of the flip-flops. Figure 2.13 suggests a trade-off between area and power consumption, which can be exploited in architectural design.

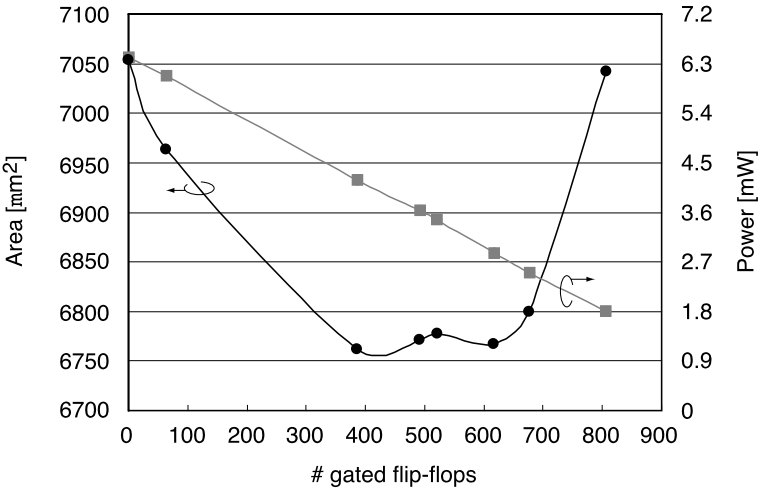


Fig. 2.13 Clock gating synthesis for wbdma [7], which contains 987 flip-flops in total

2.4.2 Dual-V_{dd}

From (2.1), it is clear that the best way to reduce dynamic power is to reduce V_{dd}. However, reducing V_{dd} comes at the cost of reduced circuit speed. The alternative approach is to use two V_{dd}'s, a higher one V_{ddh} and a lower one V_{ddl}, so that V_{ddl} is used for gates that do not affect the overall circuit speed. There are several challenges in the implementation of dual-V_{dd}:

- Layout architecture: gate placement is restricted in dual-V_{dd}, which causes an increase of area and wire length.
- Level conversion: a V_{ddl} gate needs a level converter when it drives a V_{ddh} one.
- Dual-V_{dd} allocation: automatic allocation of V_{ddh} and V_{ddl} must be done for the CAD tool.
- Selection of V_{ddl}: V_{ddh} is typically mandated by technology; V_{ddl} is thus a design parameter whose value should be carefully selected.

Layout Architecture Figure 2.14(a) shows the simplest architecture [22], where each row is dedicated to either V_{ddh} or V_{ddl} cells. Standard placement tools can be used for this architecture; once each cell is tagged for a type of row (V_{ddh} or V_{ddl}), it can be placed. The wire length typically increases substantially [23], which makes timing closure difficult to achieve; this limits the application of this architecture. Figure 2.14(b) [24] is a layout architecture, which is less constrained as far as cell placement is concerned. However, in order to minimize well isolation (see Fig. 2.15), each of the V_{ddh} and V_{ddl} cells must be grouped as much as possible; this may be performed as a refinement step after the initial placement [24], but carefully, so that initial placement is not perturbed too much. This step must be coordinated

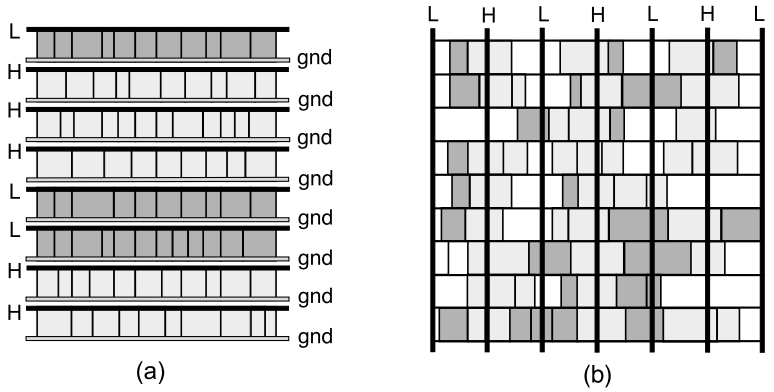


Fig. 2.14 (**a**) Row-based layout architecture and (**b**) architecture that allows less constrained placement

Fig. 2.15 Well isolation between adjacent V_{ddh} and V_{ddl} cells

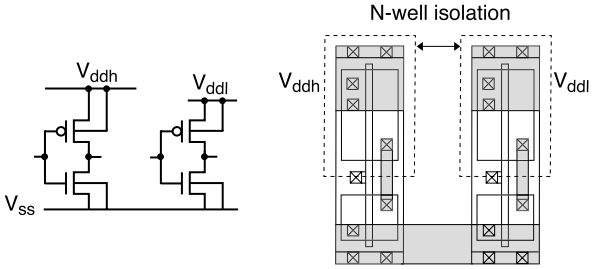
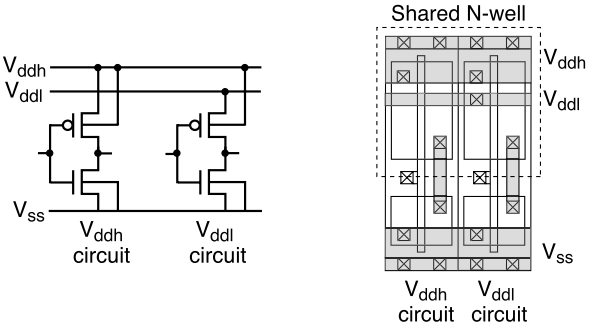


Fig. 2.16 Dual supply rail standard cells



with the topology of the generating power grid, so that each group can be placed directly below (or very close to) corresponding supply voltage rails.

Cell placement is restricted (to a different degree) in both layout architectures of Fig. 2.14. This is unavoidable as long as standard cells developed for single V_{dd} are used for dual-V_{dd} circuits. The restriction can be completely removed in the approach illustrated in Fig. 2.16 [25, 26], where each cell has dual rails for supply voltage: one for V_{ddh} and another for V_{ddl}. This of course comes at the cost of developing a custom cell library. In V_{ddh} cells, the n-well is biased to V_{ddh}, which

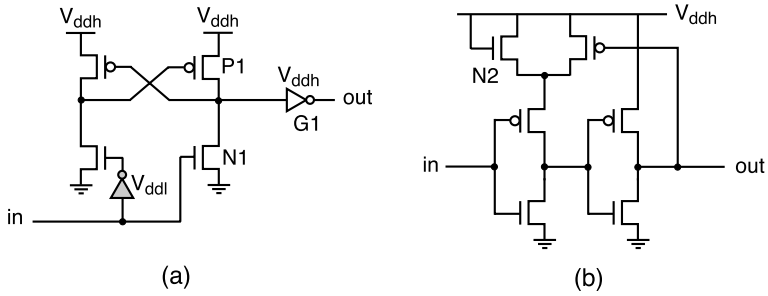


Fig. 2.17 Level converters: (**a**) pMOS cross-coupled level converter (CCLC) [22] and (**b**) single-supply level converter (SSLC) [24]

is also a voltage supply. The n-well bias of V_{ddl} cells also must be made to V_{ddh}, so that adjacent V_{ddh} and V_{ddl} cells can share their n-well for area efficiency. In this setting, the pMOS transistor of V_{ddl} cells is made slower due to its negative body bias (e.g., 18% speed degradation for $V_{ddl} = 1.2$ V with $V_{ddh} = 1.8$ V [26]); however, its subthreshold leakage is reduced substantially. The cell height increases, as it must, but only marginally (e.g., 2.7% [25]). This is because the V_{ddh} rail can be made thinner as the amount of charge it has to deliver becomes less; the V_{ddl} rail can be made even thinner, as V_{ddl} cells are typically a small proportion of the total cells.

Level Conversion The key component of dual-V_{dd} design is a level converter. If V_{ddl} is directly applied to the input of an V_{ddh} inverter, both nMOS and pMOS transistors will be turned on, if $V_{ddl} - V_{ddh}$ is smaller than the threshold voltage of pMOS; this causes huge amount of short-circuit current. Even if pMOS is not turned on, it is only weakly turned off and incurs a large amount of subthreshold current. Therefore, a level converter must be used whenever a V_{ddl} gate drives a V_{ddh} one.

Figure 2.17(a) shows a pMOS cross-coupled level converter (CCLC), which needs both a V_{ddh} and a V_{ddl} voltage supply. This can be designed as a V_{ddh} cell with V_{ddl} supplied through a pin connection; in the layout architecture shown in Fig. 2.14(a), the CCLC must be placed in V_{ddh} rows that are adjacent to a V_{ddl} row, so that a short connection to V_{ddl} can be made. Figure 2.17(b) shows a single-supply level converter (SSLC), which uses a threshold voltage drop across N2 to provide a virtual V_{ddl} to the input inverter. The SSLC is more flexible in placement than the CCLC as it uses only V_{ddh}.

In 45-nm technology with $V_{ddh} = 1.1$ V and $V_{ddl} = 0.77$ V, the CCLC and SSLC exhibit 63 ps and 49 ps of delay, respectively, with four $1\times$ inverters as load; these delay numbers represent about 1.7 and 1.3 times the V_{ddh} NAND2 delay. The area of both converters is approximately three times the NAND2 area. This suggests that the number of connections from V_{ddl} to V_{ddh} should be minimized in dual-V_{dd} circuits.

Selection of V_{ddl} It is important to decide the value of V_{ddl} for a particular V_{ddh}, which is given by the technology. Common sense tells us that there would be an

optimum value leading to minimum power consumption: if V_{ddl} is close to V_{ddh}, many gates will be allocated to V_{ddl} but the amount of power saving from each allocation will be very small; if we choose a value that is much smaller than V_{ddh}, the power saving from using V_{ddl} will be great but only a few gates will take advantage of it due to the abrupt change in delay. The important question is whether this optimum value is different for different circuits or whether there exists some universal number. We obviously prefer the latter.

It was experimentally verified [22] that the optimum V_{ddl} indeed exists at 60–70% of V_{ddh}, consistently over many circuits. This was also verified in an analytical way [27], although some approximations are involved. One work claims that optimum V_{ddl} can be brought down to 50% of V_{ddh} when dual-V_{th} is used together with dual-V_{dd} [24], because the reduced circuit speed from using lower V_{ddl} can be recovered to some extent by using dual-V_{th}.

Effectiveness of Dual-V_{dd} Several works in the literature report the effectiveness of dual-V_{dd}. In designing multimedia ASIC [22] with 0.3 µm technology, 76% of cells use V_{ddl} which yields a 47% power saving compared to a design that uses V_{ddh} alone. When this technique is applied to a microprocessor design [24] in 0.13 µm technology, the power savings is rather small, 8%. This is estimated to be due to the higher value of $V_{ddl} = 1.2$ V ($V_{ddh} = 1.5$ V) and the smaller number of V_{ddl} gates from the strict requirement on the processor design.

It is reported [28] that the layout architecture shown in Fig. 2.14(a) causes a 13% to 16% increase of the wire length depending on the placement density; Fig. 2.14(b) exhibits about a 5% increase.

2.4.3 Other Methods

To reduce the switching activity of the data path during architecture design, registers based on a dual-edge-triggered flip-flop (DETFF) can be considered. Since the DETFF is triggered at both the rising and falling edges of the clock, the clock of half the frequency can be used for the same throughput, which allows the power consumption of the clock network to be cut in half. Using latch or pulsed-latch as a register is another method to reduce clock power consumption.

A notable approach to reduce the switching activity of the data path is precomputation [29]. Two architectures to implement precomputation are shown in Fig. 2.18. In Fig. 2.18(a), the blocks g_1 and g_2 take some input bits and predict the output f; $g_1 = 1$ and $g_2 = 0$ implies $f = 1$, $g_1 = 0$ and $g_2 = 1$ implies $f = 0$, and $g_1 = 0$ and $g_2 = 0$ implies that f cannot be predicted ($g_1 = 1$ and $g_2 = 1$ are not allowed). Therefore, if $g_1 = 1$ or $g_2 = 1$ (but not both), the input bits are disabled from loading, which yields no switching activity in the combinational block; the corresponding output f is determined by g_1 or g_2. The input bits are loaded when $g_1 = g_2 = 0$ for normal computation. This architecture has a limitation due to the extra circuitry and increased critical path delay. The architecture shown in Fig. 2.18(b) is similar to

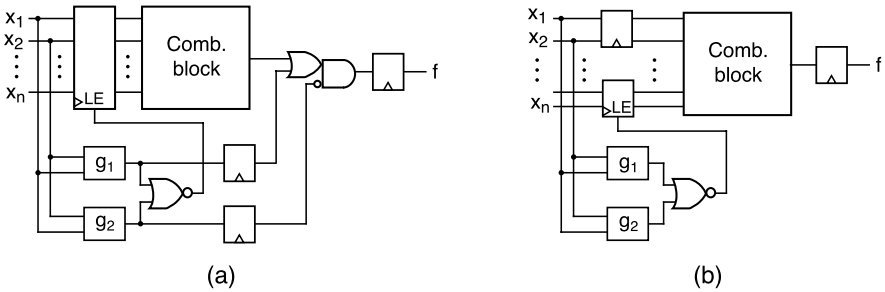


Fig. 2.18 Precomputation architectures

that of Fig. 2.18(a) except that some input bits are always loaded. Thus, if $g_1 = 1$ or $g_2 = 1$ (but not both), computation depends on only those bits that are loaded while the remaining bits are disabled from loading.

Voltage scaling is the most effective way to reduce dynamic power consumption. Designing or synthesizing a circuit such that its delay is minimized and then lowering V_{dd} until the clock period is just met could be an effective way, even though this typically comes at a cost of increased circuit area. The amount of voltage scaling is limited in ASIC design due to the fixed range of V_{dd} that is used for library characterization.

Every step of the design affects power consumption to some extent [30], e.g., technology mapping, cell and wire sizing, and floorplanning and placement. A notable power savings can be achieved if power consumption is explicitly considered during these design steps.

It is often said that there is more chance to reduce power consumption at higher abstraction levels because less is determined at that point. But the amount of power increase or decrease caused by a design decision at those levels is hard to declare in precise amounts, as discussed in Sect. 2.2.1. Any power savings at higher abstraction levels should be considered very rough and only relative.

2.5 Circuits to Reduce Static Power

The two most widely used techniques to reduce static power consumption are reviewed: power gating in Sect. 2.5.1 and body biasing in Sect. 2.5.2. Some other circuit techniques are also reviewed in Sect. 2.5.3.

2.5.1 Power Gating

Power gating[2] has become one of the most widely used circuit design techniques for reducing leakage current. Its concept, illustrated in Fig. 2.19, is essentially very

[2]This section is also available in [31].

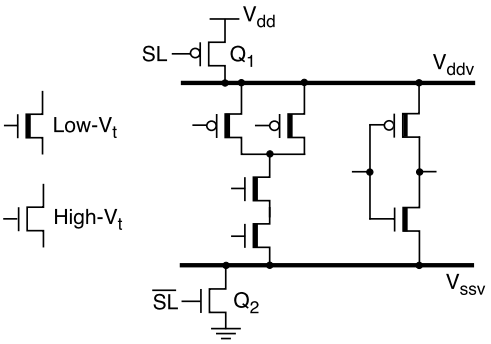


Fig. 2.19 Power gating circuit [31]. © 2010 ACM

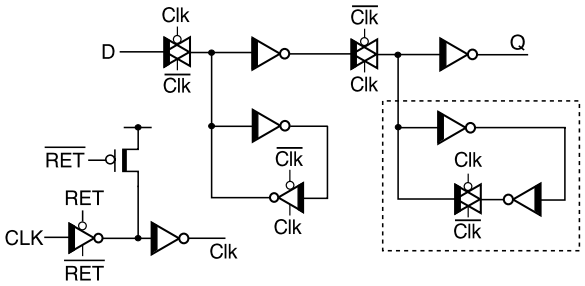


Fig. 2.20 Master-slave retention flip–flop [31]. © 2010 ACM

simple. Current switches Q_1 and Q_2 are shared by all the gates in a circuit, which are thus connected to virtual power lines V_{ddv} and V_{ssv}, instead of the real lines, V_{dd} and V_{ss}. Low-V_{th} is used within the circuit itself to achieve high performance, and high-V_{th} is used in the switches to achieve low subthreshold leakage. In active mode, SL is kept low, Q_1 and Q_2 are turned on, and V_{ddv} and V_{ssv} are maintained close to V_{dd} and V_{ss} respectively. In standby mode, SL is kept high, Q_1 and Q_2 are turned off, and V_{ddv} and V_{ssv} float; leakage from the low-V_{th} circuit is thus limited by high-V_{th} switches. These days, only one switch, a header Q_1 or a footer Q_2, is usually employed for simplicity.

Implementation Implementing power gating involves several circuit components. Virtual power lines (V_{ddv} or V_{ssv}) float once the current switches are turned off, which implies that storage elements lose their values. This can be resolved by using retention registers. There are several different implementations of retention registers; Fig. 2.20 shows one example [32]. The slave latch within the dotted box is directly connected to V_{dd} and V_{ss}, while the remainder of the circuit is connected to current switches. In sleep mode, Clk is forced to 0, which isolates the slave latch and thus allows it to retain data with RET set to 1 (0 in active mode). The latch uses a pull-up pMOS. The use of retention flip-flops invariably involves an increase in area as well as an increase in the sequencing overhead, the number of wires, and the power consumption—thus, the use of retention flip-flops should be limited. Decisions on how to provide retention are mainly made intuitively and, in any case,

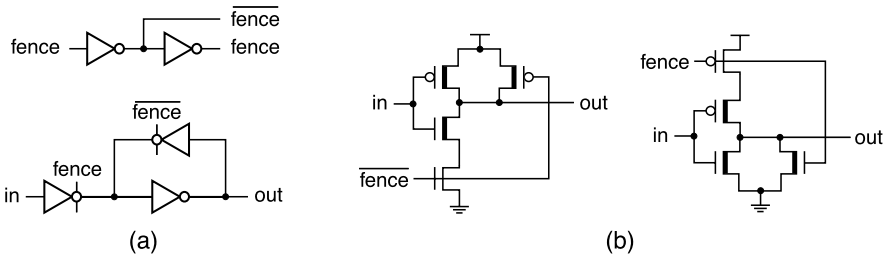


Fig. 2.21 (**a**) Output isolation circuit and (**b**) output hold circuit [31]. © 2010 ACM

retention is only required for a small fraction of the data in the system [33]. Nevertheless, some designs inherently require many retention flip-flops, such as finite state machine (FSM) controllers or register files.

After the current switches are turned off, the outputs of a circuit start to float, very slowly. This causes a large amount of short-circuit current in the blocks that are connected. A circuit that prevents the output from floating must be inserted to solve these problems. In Fig. 2.21(a) [34], out is decoupled from in once the fence is asserted; the latch then delivers the output value, which it stores in itself. However, it has limited use because it causes delay and also uses extra area and power. A simple pMOS or nMOS can be used instead, as shown in Fig. 2.21(b), if the only objective is to prevent the output from floating, and the output value itself does not need to be preserved. When a footer is used for power gating, a helper pMOS sets out to high; if a header is used, a helper nMOS sets out to low; these helper devices remain off in active mode. In practical designs, isolation circuits (Fig. 2.21(a)) are used for some outputs while the remainder simply use hold circuits (Fig. 2.21(b)).

The layout of power gating circuits using standard cells should be carefully designed. The power network should consist of three power rails V_{dd}, V_{ssv}, and V_{ss} when footers are used. But these requirements are not uniform: combinational cells need V_{dd} and V_{ssv}, retention elements need all three rails, and footer cells use V_{ssv} and V_{ss}. Similar requirements are imposed when headers are used. Figure 2.22 shows an example layout design. The V_{ssv} rails are regarded as local V_{ss} rails, while the real V_{ss} rails, on vertical layers, are regularly spaced, as shown in the figure. The required number of footer cells are now dispersed over the placement region, in locations immediately below the V_{ss} rails. The connection of V_{ss} to the retention flip-flops and isolation cells is achieved by signal routing, as we can see in the figure.

Area and Wire Length We have seen that power gating introduces extra circuitry and wires. It is important to understand the extent of the increase in area and wire length that must be tolerated in designing power gating circuits, and what factors exactly contribute to that increase; this is essential information during the early stages of design. We selected the six example circuits summarized in Table 2.2 for quantitative analysis. The first two circuits are purely combinational, while the other four are sequential. Each circuit was synthesized in 1.1 V 45-nm technology. Output

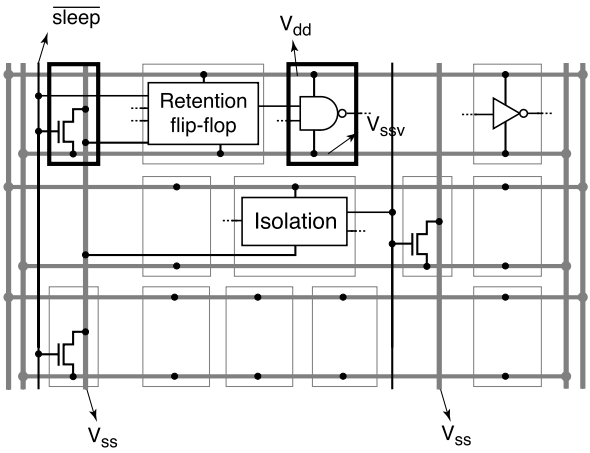


Fig. 2.22 Layout design of power gating circuit [31]. © 2010 ACM

Table 2.2 Increase in area and wire length of power gating circuits

Name	# Comb.	# FFs	# POs	Δarea (%)		Δwire length (%)	
				Header	Footer	Header	Footer
c5315	642	0	123	41	39	28	27
c6288	1211	0	32	14	10	12	18
s5378	605	176	49	41	38	38	36
b21	16441	490	22	8	6	11	8
perf_des	43044	8808	64	28	26	28	22
vga_lcd	35448	17052	108	36	34	26	24

isolation was assumed to be achieved at all primary outputs (POs) by a circuit similar to that shown in Fig. 2.21(a), and we also assumed that data would be retained at all flip-flops by circuits similar to that shown in Fig. 2.20. The ΔV across the header or footer was set to 1% of V_{dd}, and this gives us the total gate width W. If we then divide W by the gate width of a single header or footer cell, we obtain the number of header or footer cells required. The layout architecture of Fig. 2.22 was used as the basis for automatic placement and routing [35], using metal layer M3 for the V_{dd} or V_{ss} rails. However, the header or footer cells were manually placed in a regular pattern, and fixed in their locations, followed by automatic placement of the remaining cells.

The last four columns of Table 2.2 show the increase of area and wire length over the same designs without power gating. The area corresponds to the sum of the areas of all the cells. During routing, we forced the cells into about 70% of the placement region to achieve a tight placement, and metal layers up to M4 were allowed for automatic routing. In Fig. 2.23 the extra requirements of power gating are analyzed in more detail by showing the contribution of various components.

There is a fairly large increase in area (41%) when headers are introduced into c5315. This happens because it has large number of POs (considering the number of

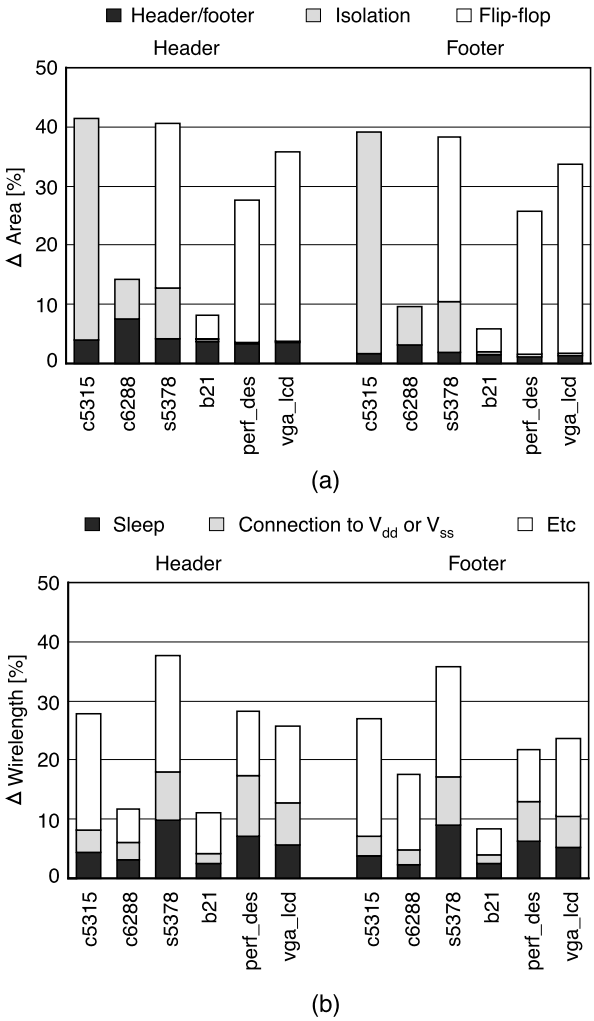


Fig. 2.23 Increase in (**a**) area and (**b**) wire length caused by power gating [31]. © 2010 ACM

combinational gates), and the same number of isolation circuits must be introduced; this is apparent from the first bar in Fig. 2.23(a). The circuit c6288 has fewer outputs, and so the increase in area is less, although 14% is still sizable; this is largely attributable to headers, as we can see in Fig. 2.23(a). The increase in area due to current switches (headers) alone is about 4% and 8% in these two circuits; this is reduced to 2% and 3% when footers are used (similar results have been reported in industrial examples [33, 36]), and the design using footers also results in lower increases in total area (39% and 10%), as shown in column 6 of Table 2.2. In the four sequential circuits, the growth in area is dominated by the retention flip-flops (see the white portions of the bars in Fig. 2.23(a)); each retention flip-flop that we used is 53% bigger than its non-power-gated counterpart. Table 2.2 shows that the increases in the area of these four sequential circuits are largely determined by the

Table 2.3 Transition energy and minimum idle time of several example circuits implemented in 65-nm technology

Name	# Comb.	# FFs	# POs	E_{tr} (pJ)	T_{min_idle} (μs)
b12	855	119	6	15.5	9.1
b13	240	53	10	9.9	17.0
irda	160	32	33	10.0	8.4
ic2	312	49	8	6.3	0.9
mc	122	17	32	9.0	11.8
wb	604	274	362	130.0	0.6

number of flip-flops, as a proportion of the total number of circuit elements. There are three components to the increase in wire length, as shown in the last two columns of Table 2.2 and in Fig. 2.23(b): extra wires for routing the `sleep` signal, the wires connecting the data retention elements to the V_{dd} or V_{ss} rails, and the increase in the lengths of all the other signal wires due to increased congestion.

The values in Table 2.2 should be considered as extreme, since data retention was assumed to be required at all POs and flip-flops. In practice, the increase in area and wire length would depend on the proportion of POs and at which data needed to be retained.

Mode Transition Turning on and off the current switches dissipates extra energy, denoted by E_{tr}. Therefore, unless a switch is turned off long enough, power gating may increase power consumption rather than decrease it. We may thus define the minimum idle period, at which power gating starts to benefit energy dissipation, T_{min_idle}. Table 2.3 shows its value for example circuits taken from International Test Conference (ITC) benchmarks and OpenCores [7], and experimentally implemented in 1.2 V 65-nm technology. It should be noted that T_{min_idle} is not directly proportional to circuit size. The value of T_{min_idle} for this particular technology is of the order of microseconds, which is equivalent to 100 clock cycles in 100 MHz and 1000 cycles at 1 GHz.

Basic power gating results in a slow wake-up process. V_{ssv} is close to V_{dd} when the footers are turned off (close to V_{ss} when the headers are turned off) during sleep mode, which implies that all the nets internal to a circuit are charged up close to V_{dd}, irrespective of their logic states. Those nets whose original logic value was 0 simultaneously start to discharge once the footers are turned on. This can cause two problems: a high current peak at V_{ss} may cause malfunctions in adjacent blocks that are still in active mode; and a large rush current combined with the parasitic of the power network and package yields a ground bounce at V_{ss}, which delays the time at which a circuit can start operation again, by increasing the wake-up delay.

Several approaches have been proposed to reduce the rush current and the wake-up delay. The VRC [37], shown in Fig. 2.24(a), produces a faster wake-up than basic power gating, since V_{ddv} and V_{ssv} are clamped to the built-in potential of a diode, and so both the charging and discharging currents during wake-up are reduced. Since the rush current is caused by turning on all the switches at the same time, a natural solution is to turn them on one by one, which can be physically implemented by

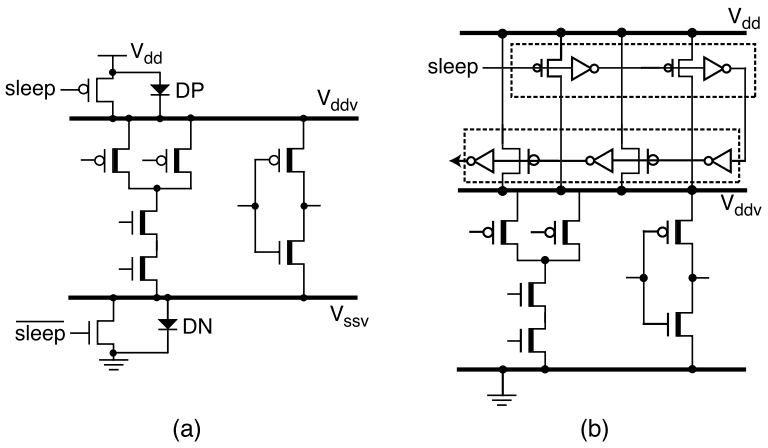


Fig. 2.24 (**a**) Virtual power/ground rail clamp (VRC) and (**b**) two-pass turn-on [31]. © 2010 ACM

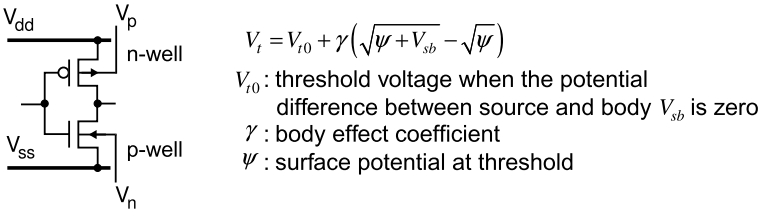


Fig. 2.25 Threshold voltage of MOS transistor

connecting the switches in a daisy-chain. The rush current is still likely to be too large unless the daisy-chain operates very slowly [38], so that the virtual rails and internal nodes have enough time to charge or discharge. Figure 2.24(b) illustrates a circuit scheme called two-pass turn-on [39], in which a header consists of two components: a series of weak switches followed by a series of strong ones. The first pass turns on the weak switches one by one, in such a way that V_{ddv} is fully restored (to a voltage close to V_{dd}) when the last switch is turned on. The limited current flow through the weak pMOS switch constrains the rush current. The weaker the first set of switches, the lower the current peak, but turn-on delay increases. In the second pass, the strong switches are turned on, and then all the switches supply the current needed in active mode.

2.5.2 Body Biasing

The threshold voltage (V_{th}) of a MOS transistor is illustrated in Fig. 2.25. Typically, the substrate (or p-well) of an nMOS is tied to V_{ss}, i.e., $V_{\mathrm{n}} = V_{\mathrm{ss}}$; similarly, the n-well of a pMOS is tied to V_{dd}, i.e., $V_{\mathrm{p}} = V_{\mathrm{dd}}$. In body biasing, V_{p} and V_{n} are

controlled (dynamically or statically), rather than being tied. In reverse body biasing (RBB), a voltage smaller than V_{ss} is applied to V_n and a voltage larger than V_{dd} is applied to V_p. As expected from Fig. 2.25, this causes V_{th} to increase since V_{sb} becomes positive; as a result, the device becomes slower and subthreshold leakage is exponentially reduced. The opposite technique is called forward body biasing (FBB).

Body biasing is a useful technique to counteract process variations. The threshold voltage always fluctuates due to imperfections in the manufacturing process and to variations in temperature and V_{dd}. When V_{dd} is high enough, this does not affect circuit delay that much since the difference between V_{dd} and V_{th}, which determines circuit delay, is large enough compared to ΔV_{th}. But, as V_{dd} becomes lower as technology is scaled down, the delay may vary a lot, which severely affects yield. In this scenario, FBB may be applied to chips that are manufactured with V_{th} larger than target, and RBB may be applied to those with unnecessarily high V_{th} thereby causing too much leakage.

Body Biasing for Low Leakage The second application of body biasing is to reduce leakage current. There are several scenarios for this purpose:

- RBB can be applied in standby mode to reduce subthreshold leakage [40].
- A circuit may be designed with V_{th} higher than its original target value, and then FBB is applied in active mode to compensate for the large initial delay, while zero body bias (ZBB) is applied in standby mode [41] to reduce the leakage.
- A circuit may be designed with V_{dd} lower than its initial target value, and then FBB is used in active mode to compensate (this helps reduce dynamic power because of the lower V_{dd} even though active leakage will be larger), while ZBB is applied in standby mode.
- A circuit may be designed so that some gates use FBB while the remainder use ZBB [41]. This technique emulates dual-V_{th} (FBB for low-V_{th} and ZBB for high-V_{th}) but without using any extra masks.

It is important to understand that RBB becomes less effective with technology scaling due to a steepening increase in junction leakage [42] with the extent of the biasing. This also implies that there is an optimal level of RBB for minimizing leakage, which is about 500 mV in sub-100 nm technology. Another technology-related problem is that RBB is impaired by increasing the within-die variation of V_{th} [43]. When body biasing is applied in dual-V_{th} circuits, care must be taken. The body effect coefficient (γ in Fig. 2.25) is different for different V_{th}, which implies that the amount of V_{th} shift is different for the same body bias.

Implementation The layout methodology of body biasing circuits is shown in Fig. 2.26, where a double-back layout pattern is employed. Standard cells are designed without taps, so that V_n and V_p are kept without any connection. A special cell, called a tap cell, must be inserted to provide the potentials for V_n and V_p, which come from body bias generator circuit. The tap cells are inserted in a regular fashion; the columns of the tap cells are separated by some predetermined distance, e.g., 50 μm [44], which is determined by the well impedance.

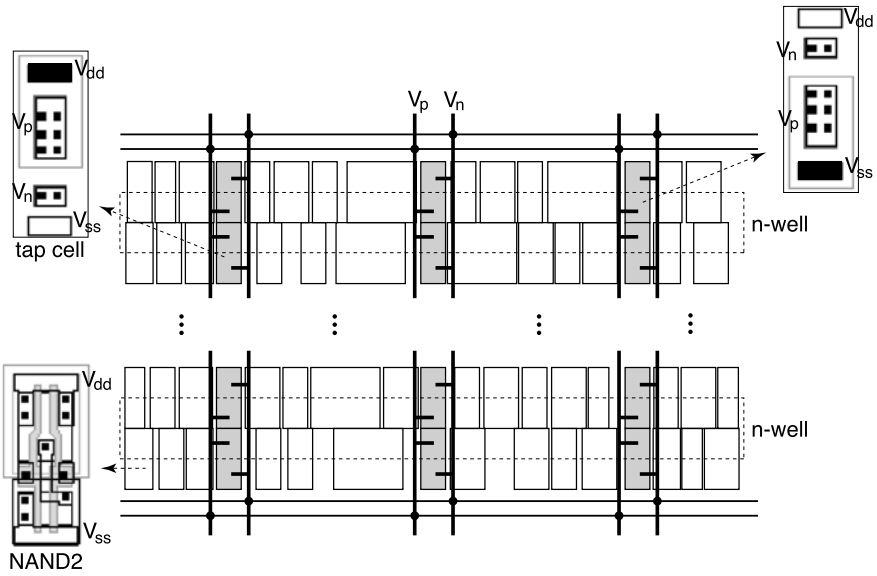


Fig. 2.26 Layout methodology of body biasing circuits [45]. © 2007 IEEE

2.5.3 Other Techniques

Many circuit design techniques have been proposed and used to control leakage. The multiple-V_{dd}, multiple-V_{th}, and multiple-gate-length approaches are static techniques; the use of two voltages, in dual-V_{dd} schemes, is most common. Dynamic techniques include input vector control (IVC), as well as power gating and body biasing. Static techniques are used during the design process with the support of a tool that determines the V_{dd} or V_{th} levels and the gate lengths of gates or blocks; the conventional synthesis-based design flow is almost unaffected. These approaches reduce leakage in both active and sleep modes, but the amount of leakage that can be saved is relatively small (e.g., 42% with dual-V_{dd} [26] and 30% using dual-gate-length [39] techniques). Dynamic techniques rely on explicit control of the leakage-reducing circuits at run time, and are typically more complicated, so that the conventional design flow has to be customized. Dynamic techniques only reduce the leakage in sleep mode, but the savings is substantial (e.g., 40× using power gating [39]). The IVC technique is an exception, in that its design is not very complicated and the leakage savings is not great, even though it must be categorized as a dynamic technique.

Multiple-V_{dd} techniques have been widely used to reduce switching power, which has a quadratic dependence on V_{dd}, but they are even more effective in reducing leakage because subthreshold and gate leakage are roughly proportional to V_{dd}^3 and V_{dd}^4, respectively [46]. The use of multiple-V_{th} configurations is prevalent in contemporary CMOS technologies; they are sometimes combined with multiple-V_{dd} or multiple-gate-length techniques.

IVC applies a predetermined input vector, called a minimum leakage vector (MLV), during sleep mode or a short idle period. Its effectiveness is largely determined by the controllability of the circuit, which is negatively correlated with logic depth. As this increases, fewer gates can usually be put in the lowest leakage state. Controllability can be improved by adding multiple control points within a circuit or by modifying some gates. Several methods have been proposed to find a good MLV, including random search and heuristics based on gate controllability or iterative dynamic programming [47], and there is an exact method using Boolean satisfiability formulation [48].

2.6 Conclusion

Architectural or system-level design involves a trade-off between one design parameter and another. Power consumption is no exception. Applying power gating, for example, helps save a substantial amount of standby leakage, but comes at a cost of extra area and wire length. Unless the trade-off is taken into account early on, an architecture may be hard to implement or may not be implementable at all. Understanding the implementation details of various low-power circuit techniques is thus very important; this has been the subject of this chapter.

References

1. Bergamaschi, R., Shin, Y., Dhanwada, N., Bhattacharya, S., Dougherty, W., Nair, I., Darringer, J., Paliwal, S.: SEAS: a system for early analysis of SoCs. In: Proc. Int. Conf. on Hardware/Software Codesign and System Synthesis, Oct. 2003, pp. 150–155 (2003)
2. Veendrick, H.: Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits. IEEE J. Solid-State Circuits 468–473 (1984)
3. Nose, K., Sakurai, T.: Analysis and future trend of short-circuit power. IEEE Trans. Comput.-Aided Des. **19**, 1023–1030 (2000)
4. Roy, K., Mukhopadhyay, S., Mahmoodi-Meimand, H.: Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. Proc. IEEE **91**(2), 305–327 (2003)
5. Narendra, S., Chandrakasan, A. (eds.): Leakage in Nanometer CMOS Technologies. Springer, Berlin (2005)
6. Ye, Y., Borkar, S., De, V.: A new technique for standby leakage reduction in high-performance circuits. In: Proc. Symp. on VLSI Circuits, June 1998, pp. 40–41 (1998)
7. OpenCores [Online]. Available: http://www.opencores.org/
8. Brand, D., Visweswariah, C.: Inaccuracies in power estimation during logic synthesis. In: Proc. Int. Conf. on Computer Aided Design, Nov. 1996, pp. 388–394 (1996)
9. Ercolani, S., Favalli, M., Damiani, M., Olivo, P., Riccó, B.: Estimate of signal probability in combinational logic networks. In: Proc. European Test Conf., Apr. 1989, pp. 132–138 (1989)
10. Borkar, S., Karnik, T., Narenda, S., Keshavarzi, A., De, V.: Parameter variations and impact on circuits and microarchitecture. In: Proc. Design Automation Conf., June 2003, pp. 338–342 (2003)
11. Su, H., Liu, F., Devgan, A., Acar, E., Nassif, S.: Full chip leakage estimation considering power supply and temperature variations. In: Proc. Int. Symp. on Low Power Electronics and Design (2003)

12. Zhao, W., Cao, Y.: New generation of predictive technology model for sub-45nm design exploration. In: Proc. Int. Symp. on Quality Electronic Design, Mar. 2006, pp. 585–590 (2006)
13. Chang, H., Sapatnekar, S.: Full-chip analysis of leakage power under process variations, including spatial correlations. In: Proc. Design Automation Conf., June 2005
14. Acar, E., Agarwal, K., Nassif, S.: Characterization of total chip leakage using inverse (reciprocal) gamma distribution. In: Proc. Int. Symp. on Circuits and Systems, May 2006
15. Rao, R., Srivastava, A., Blaauw, D., Sylvester, D.: Statistical estimation of leakage current considering inter- and intra-die process variation. In: Proc. Int. Symp. on Low Power Electronics and Design, Aug. 2003
16. Han, Y., Koren, I.: Simulated annealing based temperature aware floorplanning. J. Low Power Electron. **3**(2), 141–155 (2007)
17. Huang, W., Stan, M., Skadron, K., Sankaranarayanan, K., Ghosh, S., Velusamy, S.: Compact thermal modeling for temperature-aware design. In: Proc. Design Automation Conf., June 2004, pp. 878–883 (2004)
18. Zhan, Y., Sapatnekar, S.: A high efficiency full-chip thermal simulation algorithm. In: Proc. Int. Conf. on Computer Aided Design, Nov. 2005, pp. 635–638 (2005)
19. Chinnery, D., Keutzer, K.: Closing the power gap between ASIC and custom: an ASIC perspective. In: Proc. Design Automation Conf., June 2005, pp. 275–280 (2005)
20. Arbel, E., Eisner, C., Rokhlenko, O.: Resurrecting infeasible clock-gating functions. In: Proc. Design Automation Conf., July 2009, pp. 160–165 (2009)
21. Synopsys. IC Compiler User Guide, Dec. 2008
22. Usami, K., Igarashi, M., Minami, F., Ishikawa, T., Kanzawa, M., Ichida, M., Nogami, K.: Automated low-power technique exploiting multiple supply voltages applied to a media processor. IEEE J. Solid-State Circuits **33**(3), 463–472 (1998)
23. Shin, I., Paik, S., Shin, Y.: Register allocation for high-level synthesis using dual supply voltages. In: Proc. Design Automation Conf., July 2009, pp. 937–942 (2009)
24. Puri, R., Stok, L., Cohn, J., Kung, D., Pan, D., Sylvester, D., Srivastava, A., Kulkarni, S.: Pushing ASIC performance in a power envelope. In: Proc. Design Automation Conf., June 2003, pp. 788–793 (2003)
25. Wang, J., Shieh, S., Wang, J., Yeh, C.: Design of standard cells used in low-power ASIC's exploiting the multiple-supply-voltage scheme. In: Proc. ASIC Conf., Sept. 1998, pp. 119–123 (1998)
26. Shimazaki, Y., Zlatanovici, R., Nikoli, B.: A shared-well dual-supply-voltage 64-bit ALU. In: Proc. IEEE Int. Solid-State Circuits Conf., Feb. 2003, pp. 104–105 (2003)
27. Kuroda, T., Hamada, M.: Low-power CMOS digital design with dual embedded adaptive power supplies. IEEE J. Solid-State Circuits **35**(4), 652–655 (2000)
28. Shin, I., Paik, S., Shin, D., Shin, Y.: HLS-dv: a high-level synthesis framework for dual-V_{dd} architectures. IEEE Trans. Very Large Scale Integr. (2011)
29. Alidina, M., Monteiro, J., Devadas, S., Ghosh, A., Papaefthymiou, M.: Precomputation-based sequential logic optimization for low power. IEEE Trans. Very Large Scale Integr. **2**(4), 426–436 (1994)
30. Chinnery, D., Keutzer, K. (eds.): Closing the Power Gap Between ASIC & Custom. Springer, Berlin (2007)
31. Shin, Y., Seomun, J., Choi, K.-M., Sakurai, T.: Power gating: circuits, design methodologies, and best practice for standard-cell VLSI designs. ACM Trans. Des. Autom. Electron. Syst. **15**(4), 28:1–28:37 (2010)
32. Mair, H., Wang, A., Gammie, G., Scott, D., Royannez, P., Gururajarao, S., Chau, M., Lagerquist, R., Ho, L., Basude, M., Culp, N., Sadate, A., Wilson, D., Dahan, F., Song, J., Carlson, B., Ko, U.: A 65-nm mobile multimedia applications processor with an adaptive power management scheme to compensate for variations. In: Proc. Symp. on VLSI Circuits, June 2007, pp. 224–225 (2007)
33. Lueftner, T., et al.: A 90-nm CMOS low-power GSM/EDGE, multimedia-enhanced baseband processor with 380-MHz ARM926 core and mixed-signal extensions. IEEE J. Solid-State Circuits **42**(1), 134–144 (2007)

34. Won, H.-S., Kim, K.-S., Jeong, K.-O., Park, K.-T., Choi, K.-M., Kong, J.-T.: An MTCMOS design methodology and its application to mobile computing. In: Proc. Int. Symp. on Low Power Electronics and Design, Aug. 2003, pp. 110–115 (2003)
35. Synopsys. Astro User Guide, Mar. 2007
36. Kanno, Y., et al.: Hierarchical power distribution with power tree in dozens of power domains for 90-nm low-power multi-CPU SoCs. IEEE J. Solid-State Circuits **42**(1), 74–83 (2007)
37. Kumagai, K., Iwaki, H., Yoshida, H., Suzuki, H., Yamada, T., Kurosawa, S.: A novel powering-down scheme for low Vt CMOS circuits. In: Proc. Symp. on VLSI Circuits, June 1998, pp. 44–45 (1998)
38. Shi, K., Howard, D.: Challenges in sleep transistor design and implementation in low-power designs. In: Proc. Design Automation Conf., July 2006, pp. 113–116 (2006)
39. Royannez, P., Mair, H., Dahan, F., Wagner, M., Streeter, M., Bouetel, L., Blasquez, J., Clasen, H., Semino, G., Dong, J., Scott, D., Pitts, B., Raibaut, C., Ko, U.: 90 nm low leakage SoC design techniques for wireless applications. In: Proc. IEEE Int. Solid-State Circuits Conf., Feb. 2005, pp. 138–139 (2005)
40. Seta, K., Hara, H., Kuroda, T., Kakumu, M., Sakurai, T.: 50% active-power saving without speed degradation using standby power reduction (SPR) circuit. In: Proc. IEEE Int. Solid-State Circuits Conf., Feb. 1995, pp. 318–319 (1995)
41. Narendra, S., Keshavarzi, A., Bloechel, B., Borkar, S., De, V.: Forward body bias for microprocessors in 130-nm technology generation and beyond. IEEE J. Solid-State Circuits **38**(5), 696–701 (2003)
42. Keshavarzi, A., Narendra, S., Borkar, S., Hawkins, C., Roy, K., De, V.: Technology scaling behavior of optimum reverse body bias for standby leakage power reduction in CMOS IC's. In: Proc. Int. Symp. on Low Power Electronics and Design, Aug. 1999, pp. 252–254 (1999)
43. Narenda, S., Antoniadis, D., De, V.: Impact of using adaptive body bias to compensate die-to-die Vt variation on within-die Vt variation. In: Proc. Int. Symp. on Low Power Electronics and Design, Aug. 1999, pp. 229–232 (1999)
44. Clark, L., Morrow, M., Brown, W.: Reverse-body bias and supply collapse for low effective standby power. IEEE Trans. Very Large Scale Integr. **12**(9), 947–956 (2004)
45. Choi, B., Shin, Y.: Lookup table-based adaptive body biasing of multiple macros. In: Proc. IEEE Int. Symp. on Quality Electronic Design, ISQED, Mar. 2007, pp. 533–538 (2007)
46. Krishnamurthy, R., Alvandpour, A., De, V., Borkar, S.: High-performance and low-power challenges for sub-70nm microprocessor circuits. In: Proc. Custom Integrated Circuits Conf., May 2002, pp. 125–128 (2002)
47. Cheng, L., Deng, L., Chen, D., Wong, M.: A fast simultaneous input vector generation and gate replacement algorithm for leakage power reduction. In: Proc. Design Automation Conf., July 2006, pp. 117–120 (2006)
48. Aloul, F., Hassoun, S., Sakallah, K., Blaauw, D.: Robust SAT-based search algorithm for leakage power reduction. Lect. Notes Comput. Sci. **2451**, 253–274 (2002)

http://www.springer.com/978-94-007-1678-0

Energy-Aware System Design
Algorithms and Architectures
Kyung, C.; Yoo, S. (Eds.)
2011, IX, 291 p., Hardcover
ISBN: 978-94-007-1678-0